Un seul serveur vous manque, et tout est découplé !

Thomas Bonald[1], Céline Comte[21] et Fabien Mathieu[2] †

[1] *Télécom ParisTech, Université Paris-Saclay, France*
[2] *Nokia Bell Labs, France*

Les infrastructures de cloud computing reposent sur des grappes de serveurs qui se partagent des requêtes. Leur dimensionnement nécessite de prédire leurs performances. L'un des principaux défis consiste à tenir compte des interactions complexes entre serveurs lorsqu'ils sont mis en commun pour traiter des requêtes en parallèle, en intégrant certaines contraintes comme la localité des données qui restreignent les affectations possibles. Pour les analyser, nous représentons ces contraintes par un graphe d'affectation entre les requêtes et les serveurs ; les ressources sont partagées selon l'équité équilibrée, qui a l'avantage de rendre les performances du système insensibles à la distribution de la taille des requêtes. Notre principale contribution est une nouvelle approche récursive pour calculer les métriques de performance, consistant à décomposer le système en *éteignant* les serveurs les uns après les autres. Même si la complexité des formules obtenues peut être exponentielle en la taille de la grappe dans le pire cas, nous illustrons leur intérêt pratique en identifiant de vastes familles de structures pour lesquelles elle devient polynomiale. Ceci étend considérablement l'ensemble des systèmes pour lesquels des métriques de performance explicites sont accessibles.

Une version longue de cet article a été publiée dans POMACS [BCM17] et sera présentée à *SIGMETRICS 2018*.

■Équité équilibrée, parallélisme, évaluation de performance

1 Grappe de serveurs sous l'équité équilibrée

Nous considérons un ensemble $\mathcal{K}$ de K serveurs, appelé grappe, recevant un ensemble $\mathcal{I}$ de I classes de requêtes. Les requêtes de chaque classe arrivent selon un processus de Poisson et ont des tailles i.i.d. Chaque requête quitte le système dès que son service est terminé. Pour chaque $i \in \mathcal{I}$, on note λ_i l'intensité de trafic de la classe i, définie comme la quantité moyenne de travail apportée par les requêtes de cette classe par unité de temps. Pour chaque $k \in \mathcal{K}$, la capacité de service du serveur k est notée μ_k.

La classe d'une requête définit l'ensemble des serveurs auxquels elle est affectée. Elle peut être déterminée par des contraintes pratiques comme la localité des données ou bien résulter d'une répartition statique de la charge (des exemples seront donnés en partie 3). Pour chaque $k \in \mathcal{K}$, on note $\mathcal{I}_k \subset \mathcal{I}$ l'ensemble des classes de requêtes que le serveur k peut traiter. La contrainte des serveurs mis en grappe peut être représentée par un graphe biparti entre classes et serveurs, comme illustré en FIGURE 1. On suppose sans perte de généralité que les requêtes de chaque classe sont affectées à au moins un serveur et que chaque serveur se voit affecter les requêtes d'au moins une classe.

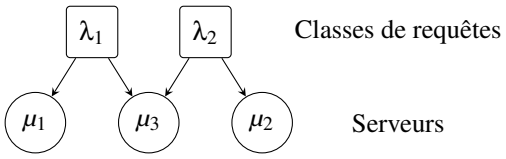

FIGURE 1: Un graphe d'affectation entre trois serveurs et deux classes de requêtes. Les serveurs 1 et 2 sont respectivement dédiés aux classes 1 et 2, et le serveur 3 est partagé. On a $\mathcal{I}_1 = \{1\}$, $\mathcal{I}_2 = \{2\}$ et $\mathcal{I}_3 = \{1,2\}$.

†Les auteurs sont membres du LINCS, voir www.lincs.fr.

On suppose vérifiée la condition suivante de stabilité, qui impose que tout ensemble de serveurs ait une capacité suffisante pour traiter les requêtes exclusivement affectables à des serveurs de cet ensemble :

$$\sum_{i\in\mathcal{I}\setminus\bigcup_{k\in\mathcal{K}\setminus\mathcal{L}}\mathcal{I}_k}\lambda_i<\sum_{k\in\mathcal{L}}\mu_k,\quad\forall\mathcal{L}\subset\mathcal{K}.$$

Cette condition, nécessaire pour garantir la stabilité, sera suffisante sous l'équité équilibrée [BP03, SdV15].

Allocation des ressources Les ressources des serveurs sont partagées selon l'équité équilibrée [BP03]. Celle-ci généralise la discipline de service processor-sharing dans le sens où toutes les requêtes sont servies simultanément, celles *d'une même classe* recevant le même taux de service. Le partage des ressources est défini globalement à l'échelle de la grappe toute entière, comme décrit dans [SdV15].

Lorsqu'elles sont calculables, les métriques de performance sous l'équité équilibrée donnent des intuitions robustes sur les performances réelles du système. L'équité équilibrée satisfait en effet la propriété d'insensibilité, qui implique que les performances sous les hypothèses données plus haut sont identiques à celles obtenues lorsque les tailles des requêtes sont distribuées selon la loi exponentielle. L'état du système (décrit par le nombre de requêtes de chaque classe présentes) définit alors un processus de Markov réversible. Par ailleurs, il a été prouvé dans [SdV15] que l'équité équilibrée est Pareto-efficace dans les grappes de serveurs. Cela signifie en pratique que la capacité de chaque serveur est entièrement utilisée dès qu'une requête lui est affectée.

Performance D'après la loi de Little, prédire les performances du système revient à calculer l'espérance L du nombre de requêtes présentes dans le système en régime stationnaire. On peut montrer que, sous l'équité équilibrée, la valeur de L découle directement de la probabilité ψ que le système stationnaire soit vide, elle-même égale à l'inverse de la constante de normalisation. Le calcul de cette dernière n'est pas immédiat puisque l'espace d'états est infini.

Ce problème peut être contourné de différentes manières. Dans [BV04, SdV15], une étape d'agrégation permet de se ramener à un espace d'états fini dont la taille est exponentielle en le nombre de classes. Des conditions suffisantes pour obtenir une taille polynomiale ont été identifiées dans [BCSd17] mais elles s'appliquent difficilement aux grappes de serveurs. Notre approche est différente. Elle généralise les résultats présentés dans [GHBSW$^+$17, GHBHR17], qui sont des cas particuliers de notre théorème principal.

2 Calcul récursif des métriques de performance

Notre résultat repose sur la remarque suivante, qui découle de la réversibilité du processus de Markov défini par l'état du système dans le cas exponentiel :

> La distribution stationnaire conditionnelle du système sachant qu'un serveur donné est inactif est égale à la distribution stationnaire du sous-système où aucun trafic n'est généré par les classes de requêtes affectées à ce serveur.

Pour chaque serveur $k\in\mathcal{K}$, on a $\psi=\psi_k\psi_{|-k}$, où ψ_k est la probabilité que le serveur k soit inactif et $\psi_{|-k}$ est la probabilité conditionnelle que le système soit vide sachant que le serveur k est inactif. La remarque ci-dessus montre que $\psi_{|-k}$ est aussi la probabilité que le sous-système où les classes de $\mathcal{I}_k$ ne génèrent aucun trafic soit vide. Cette égalité ne permet pas de calculer directement ψ (on ne connaît pas ψ_k), mais elle sera utilisée dans la preuve du théorème 1 pour exprimer ψ en fonction des probabilités $\psi_{|-k}$ pour $k\in\mathcal{K}$.

De même, si l'on note $L_{|-k}$ l'espérance conditionnelle du nombre de requêtes dans le système sachant que le serveur k est inactif, alors $L_{|-k}$ est aussi l'espérance du nombre de requêtes dans le sous-système où les classes de $\mathcal{I}_k$ ne génèrent aucun trafic.

Le théorème suivant donne une méthode effective pour calculer les métriques globales de performance par récurrence. Dans [BCM17], nous obtenons un résultat similaire pour évaluer les performances par classe.

Théorème 1. *Soit* $\rho=\frac{\sum_{i\in\mathcal{I}}\lambda_i}{\sum_{k\in\mathcal{K}}\mu_k}$ *la charge totale dans le système. On a*

$$\psi=(1-\rho)\frac{\sum_{k\in\mathcal{K}}\mu_k}{\sum_{k\in\mathcal{K}}\frac{\mu_k}{\psi_{|-k}}}\qquad\text{et}\qquad L=\frac{\rho}{1-\rho}+\frac{1}{1-\rho}\frac{\sum_{k\in\mathcal{K}}\mu_k\psi_kL_{|-k}}{\sum_{k\in\mathcal{K}}\mu_k}.$$

Démonstration. On se contente de prouver la formule donnant ψ. L'équation de conservation stipule qu'à l'état stationnaire, la quantité de travail qui entre est égale à la quantité de travail effectivement fournie :

$$\sum_{i\in\mathcal{I}} \lambda_i = \sum_{k\in\mathcal{K}} \mu_k(1-\psi_k), \qquad \text{soit} \qquad \sum_{k\in\mathcal{K}} \mu_k\psi_k = \sum_{k\in\mathcal{K}} \mu_k - \sum_{i\in\mathcal{I}} \lambda_i.$$

On obtient la formule annoncée pour ψ en remplaçant ψ_k par $\frac{\psi}{\psi_{|-k}}$ et en réarrangeant les termes. □

On distingue deux contributions dans les expressions de ψ et L. La première donne les performances dans une grappe où toutes les requêtes sont affectées à tous les serveurs (i.e., avec un graphe d'affectation biparti complet). La seconde donne le surcoût dû au partage incomplet des ressources. Pour ψ, ce surcoût est égal à la moyenne harmonique des probabilités $\psi_{|-k}$, $k \in \mathcal{K}$, pondérées par les capacités μ_k, $k \in \mathcal{K}$, des serveurs.

3 Applications

Le théorème 1 permet de calculer récursivement ψ et L en découplant le système progressivement. Le cas de base correspond à une grappe sans trafic, vide avec probabilité 1. Sans autre hypothèse, il est nécessaire de réaliser $O(I+K)$ opérations pour chacun des 2^K ensembles possibles de serveurs inactifs, ce qui donne une complexité $O\left((I+K)2^K\right)$. Cependant, la complexité peut être rendue polynomiale en exploitant des symétries ou des propriétés topologiques, comme dans les deux scénarios suivants.

Répartition statique de la charge Dans [GHBSW+17], Gardner *et al.* étudient un système où chaque requête est affectée à d serveurs choisis au hasard de façon uniforme dans une grappe homogène de taille K, et donnent une formule de complexité $O(K)$ pour évaluer les performances. Ce système peut être décrit dans notre modèle en considérant les $\binom{K}{d}$ affectations possibles comme autant de classes de requêtes. Tous les sous-systèmes contenant le même nombre de serveurs inactifs sont équivalents, de sorte qu'il y a seulement K sous-systèmes à considérer au lieu de 2^K. Le théorème 1 redonne immédiatement les résultats de [GHBSW+17]. Ces résultats peuvent être généralisés en ajoutant une hétérogénéité limitée (du côté des serveurs et/ou des requêtes) tout en gardant une complexité polynomiale en fonction du nombre de serveurs.

Affectations locales Un autre cas intéressant est celui où les serveurs sont ordonnés le long d'une ligne virtuelle et où chaque classe est associée à un intervalle de serveurs. Une telle ligne privée d'un serveur se divise en (au plus) deux sous-lignes qui *évoluent de façon indépendante*. Il n'est donc plus nécessaire de considérer chacun des 2^K sous-ensembles de serveurs possibles, mais seulement les $O(K^2)$ sous-lignes. Les formules obtenues ont une complexité en $O(K^3)$. Les résultats s'appliquent en particulier aux structures en classes imbriquées proposées dans [GHBHR17], et s'étendent aisément à une topologie en anneau. La notion d'intervalle de serveurs permet de prendre en compte l'éventuelle proximité physique nécessaire pour faciliter le parallélisme. Elle apparaît aussi naturellement si les requêtes sont affectées aux serveurs selon une table de hachage distribuée munie d'une topologie en anneau, par exemple de type Chord [SMK+01].

Exemple numérique Pour illustrer l'intérêt pratique de nos résultats, imaginons qu'un opérateur souhaite implanter une répartition statique de la charge pour des requêtes parallélisables sur d serveurs dans une grappe homogène de taille $K = 100$. Lorsqu'une requête arrive, elle est affectée à un ensemble de d serveurs choisi au hasard de façon uniforme parmi un ensemble de possibilités. L'opérateur compare plusieurs modes de répartition qui restreignent les affectations possibles : *global* (n'importe quel ensemble de d serveurs), en *ligne* (un intervalle de d serveurs dans une topologie en ligne) et en *anneau* (un intervalle de d serveurs dans une topologie en anneau). Ces scénarios sont des cas particuliers des structures étudiées plus haut.

La FIGURE 2 donne un comparatif du taux moyen de service, défini comme la vitesse moyenne à laquelle les requêtes sont traitées, et calculé en temps polynomial grâce au théorème 1. Une affectation globale semble toujours plus efficace qu'une affectation locale (en ligne ou en anneau). Deux principaux facteurs expliquent ce phénomène. D'une part, restreindre les affectations à des ensembles de serveurs consécutifs diminue le nombre d'affectations possibles et augmente la probabilité que deux requêtes partagent un grand nombre de serveurs. Ceci crée un *prix de localité*, particulièrement visible à charge moyenne (figure de gauche). D'autre part, la structure en ligne souffre d'un désavantage supplémentaire : l'*hétérogénéité*. Les

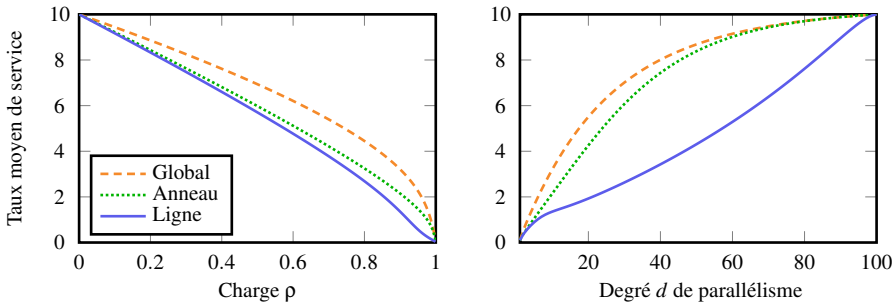

FIGURE 2: Comparaison des répartitions globale et locales (valeurs par défaut : $K = 100$, $d = 10$, $\rho = 0.9$)

serveurs centraux reçoivent plus de requêtes que les serveurs en périphérie, de sorte que les requêtes affectées à ces serveurs ont des performances moindres. Si cette hétérogénéité n'impacte pas la stabilité, elle génère une dégradation des performances qui est particulièrement sensible au choix de d (figure de droite).

4 Conclusion

Nous avons proposé une nouvelle formule pour calculer les performances d'une grappe de serveurs dont les ressources sont allouées selon l'équité équilibrée. L'ingrédient clé consiste à évaluer la probabilité qu'un serveur soit inactif en comparant le comportement du système avec et sans ce serveur. Bien que de complexité exponentielle dans le pire cas, notre formule fournit une approche unifiée qui simplifie l'étude de nombreux systèmes intéressants en pratique. Dans l'avenir, nous espérons pouvoir identifier d'autres structures de grappes dont l'étude est rendue possible par notre formule, par exemple en exploitant les critères de symétrie et d'indépendance déjà utilisés. Nous souhaitons aussi utiliser ce résultat pour mieux comprendre l'impact du graphe d'affectation sur les performances.

Références

[BCM17] T. Bonald, C. Comte, and F. Mathieu. Performance of balanced fairness in resource pools : A recursive approach. *Proc. ACM Meas. Anal. Comput. Syst.*, 1(2) :41 :1–41 :25, Dec 2017.

[BCSd17] T. Bonald, C. Comte, V. Shah, and G. de Veciana. Poly-symmetry in processor-sharing systems. *Queueing Systems*, 86(3-4) :327–359, August 2017.

[BP03] T. Bonald and A. Proutière. Insensitive Bandwidth Sharing in Data Networks. *Queueing Systems*, 44(1) :69–100, May 2003.

[BV04] T. Bonald and J. Virtamo. Calculating the flow level performance of balanced fairness in tree networks. *Performance Evaluation*, 58(1) :1–14, October 2004.

[GHBHR17] K. Gardner, M. Harchol-Balter, E. Hyytiä, and R. Righter. Scheduling for efficiency and fairness in systems with redundancy. *Performance Evaluation*, 116 :1–25, November 2017.

[GHBSW+17] K. Gardner, M. Harchol-Balter, A. Scheller-Wolf, M. Velednitsky, and S. Zbarsky. Redundancy-*d* : The Power of *d* Choices for Redundancy. *Operations Research*, 65(4) :1078–1094, April 2017.

[GZD+15] K. Gardner, S. Zbarsky, S. Doroudi, M. Harchol-Balter, and E. Hyytia. Reducing latency via redundant requests : Exact analysis. *SIGMETRICS Perf. Eval.*, 43(1) :347–360, 2015.

[SdV15] V. Shah and G. de Veciana. High-Performance Centralized Content Delivery Infrastructure : Models and Asymptotics. *Transactions on Networking*, 23(5) :1674–1687, 2015.

[SMK+01] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord : A scalable peer-to-peer lookup service for Internet applications. *SIGCOMM Comput. Commun. Rev.*, 31(4) :149–160, 2001.